

PL/M-80 COMPILER CP/M 3.0 --- SHOW 3.1

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SHOW

OBJECT MODULE PLACED IN SHOW.OBJ

COMPILER INVOKED BY: PLM80 SHOW.PLM DEBUG OPTIMIZE PAGEWIDTH(132)

\$ TITLE('CP/M 3.0 --- SHOW 3.1')

/*

Revised:

Oct 82 by Phillip Balma

14 Sept 81 by Doug Huskey

*/

*** SHOW ***

```
1      show;
      do;
2 1      declare
          mpw          literally '30h';
3 1      declare plw          label public;
4 1      declare copyright(*) byte data
          (' Copyright (c) 1982, Digital Research ');
5 1      declare verdate(*)    byte data('10/27/82'),
          version(*)          byte data('Show 3.1');
```

/*

copyright(c) 1975, 1976, 1977, 1978, 1979, 1980, 1981,1982
digital research
box 579
pacific grove, ca
93950

*/

/* modified 10/30/78 to fix the space computation */
/* modified 01/28/79 to remove despool dependencies */
/* modified 07/26/79 to operate under cp/m 2.0 */
/* modified 01/20/80 by Thomas Rolander */
/* show created 05/19/81 */
/* modified 7/82 to add new options parser, # dir FCB's left, new DISK option,
of files by Phillip Balma */
/* added paging, # SFCB's Phillip Balma*/

CP/M-PLUS V3.0

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # CP3-135

```

*** DISK INTERFACE ***

```

* * * * *

```

6 1 declare
    dcnt          byte,
    anything      byte,
    dirbuf(128)   byte;

7 1 declare
    line$page     byte,
    line$out      byte,
    drives(16)    byte,
    drive         byte,
    all           byte initial(0),
    once$only     byte initial(0),
    done$drive(16) byte initial(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),

    PAGE          byte initial(0),
    NONBANK       byte initial(0),

    user(16)      byte,          /* any files in user i? */
    used(16)      address,        /* # files in user i */
    free$dir      address,        /* # free directories */
    nSFCB         address,        /* # SFCB's */

    SCBPB         structure(
        where     byte,
        set       byte,
        value     address) initial(0,0,0),

    ERRORM(*)     byte data('ERROR: ',0),
    input(*)      byte data('INPUT: ',0),
    eoption(*)    byte data('OPTION: ',0),
    dirdrive(*)   byte data('DRIVE: ',0),

    err$unrecopt(*) byte data('Unrecognized Option.',0),
    err$unrecd(*)  byte data('Unrecognized drive.',0),
    err$version(*) byte data('Requires CP/M 3 or higher.',0),
    err$no label(*) byte
        data('No directory label exists on drive ',0),
    err$input(*)  byte data('Unrecognized input.',0),

    opt$dir       byte data(1),
    opt$drive     byte data(2),
    opt$label     byte data(3),
    opt$space     byte data(0),
    opt$user      byte data(4),
    opt$page      byte data(6),          /*rel to 1 */
    opt$nopage    byte data(7);

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. #

3 1 declare

```
dirs(*) byte data
      ('A:0B:0C:0D:0E:0F:0G:0H:0I:0J:0K:0L:0M:0N:0',
       '0:0P:',0ffh),
```

```

options(*) byte data('SPACE0DIRECT0RY0DRIVE0LABEL0USER0',
                    'PAGE0NOPAGE',0ffh),

off$dirs(*) byte data(0,3,6,9,12,15,18,21,24,27,30,33,36,39,42,
                    45,47),
off$opt(*) byte data(0,6,16,23,29,35,40,46),

end$list      byte data (0ffh),
end$of$string byte data (0),

delimiters(*) byte data (0,['[]=, ;;<>Z\!'()/#!@&+-*?',0,0ffh),
SPACE        byte data(5), /* index into delim to space */
EOS          byte data(25),
COMMA        byte data(4),
COLON        byte data(6),
LBRACKET     byte data(1),
RBRACKET     byte data(2),

opt$map(21)   structure ( option(5) byte),

j            byte initial(0),
buf$ptr      address,
opt$index    byte,
endbuf       byte,
delimiter    byte;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

\$ eject

9 1 declare

```

maxb      address external,      /* addr field of jmp BDOS */
fcb(33)   byte external,         /* default fcb */
buff(128) byte external,         /* default buffer */
fcba      literally '.fcb',      /* default fcb */
dolla     literally '.fcb(6dh-5ch)', /* $ position */
rreca     literally '.fcb(7dh-5ch)', /* ran rcd 7d,7e,7f */
rreco     literally '.fcb(7fh-5ch)', /* ran overflow */
sectorlen literally '128',        /* sector length */
rrec      address at(rreca),      /* random record address */
rovf      byte at(rreco),        /* overflow on getfile */
doll      byte at(dolla),        /* dollar parameter */
user$code byte,                  /* current user code */
cversion  address,               /* BDOS version $ */
cdisk     byte,                  /* current disk */

```

/* function call 32 returns the address of the disk parameter

block for the currently selected disk, which consists of:

```

scptrk    (2 by) number of sectors per track
blkshf    (1 by) log2 of blocksize (2**blkshf=blksize)
blkmsk    (1 by) 2**blkshf-1
extmsk    (1 by) logical/physical extents
maxall    (2 by) max alloc number
dirmax    (2 by) size of directory-1
alloc     (2 by) reservation bits for directory
chksize   (2 by) size of checksum vector
offset    (2 by) offset for operating system
psh       (1 by) log2 of physical record size (2**psh * 128 = size)
psm       (1 by) 2**psh - 1

```

*/

```

dpba      address,               /* disk parameter block address */
dpb       based dpba structure(
spt       address,
bls       byte,
bms       byte,
exm       byte,
mxm       address,
dmx       address,
dbl       address,
cks       address,
ofs       address,
psh       byte,
psm       byte),

```

```

scptrk    literally 'dpb.spt',
blkshf    literally 'dpb.blk',
blkmsk    literally 'dpb.bms',
extmsk    literally 'dpb.exm',
maxall    literally 'dpb.mxm',
dirmax    literally 'dpb.dmx',
dirblk    literally 'dpb.dbl',
chksize   literally 'dpb.cks',
offset    literally 'dpb.ofs',
physhf    literally 'dpb.psh',

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

phymask literally 'dph.psm';

```

10 1  boot: procedure external;
      /* reboot */
11 2  end boot;

12 1  mon1: procedure(f,a) external;
13 2  declare f byte, a address;
14 2  end mon1;

15 1  mon2: procedure(f,a) byte external;
16 2  declare f byte, a address;
17 2  end mon2;

18 1  declare mon3 literally 'mon2a';

19 1  mon3: procedure(f,a) address external;
20 2  declare f byte, a address;
21 2  end mon3;

22 1  declare alloca address,
      /* alloca is the address of the disk allocation vector */
      alloc based alloca (1024) byte; /* allocation vector */

23 1  declare
      true      literally '1',
      false     literally '0',
      forever   literally 'while true',
      lit       literally 'literally',
      proc      literally 'procedure',
      dcl       literally 'declare',
      addr      literally 'address',
      ctlc      literally '3',
      cr        literally '13',
      lf        literally '10';

24 1  printchar: procedure(char);
25 2  declare char byte;
26 2  call mon1(2,char);
27 2  end printchar;

28 1  printb: procedure;
      /* print blank character */
29 2  call printchar(' ');
30 2  end printb;

31 1  printx: procedure(a);
32 2  declare a address;
33 2  declare s based a byte;

34 2  do while s <> 0;
35 3  call printchar(s);
36 3  a = a + 1;
37 3  end;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

38 2    end printx;

39 1    break: procedure byte;
40 2        return mon2(11,0);    /* console ready */
41 2    end break;

42 1    crlf2: procedure;
43 2        call printchar(cr);
44 2        call printchar(lf);
45 2    end crlf2;

46 1    terminate: procedure;
47 2        call crlf2;
48 2        call mon1 (0,0);
49 2    end terminate;    /* system reset */

50 1    crlf: procedure;
51 2        if PAGE then do;
53 3            line$out = line$out + 1;
54 3            if line$out + 2 > line$page then do;
56 4                call crlf2;
57 4                call crlf2;
58 4                call printx(,('Press RETURN to continue.',0));
59 4                do while not break;    /* wait until a console break*/
60 5                    end;
61 4                if mon2(1,0) = ctlc then call terminate;
63 4                line$out = 1;
64 4                call crlf2;
65 4                end;
66 3            end;

67 2        call crlf2;

68 2    end crlf;

69 1    print: procedure(a);
70 2        declare a address;
    /* print the string starting at address a until the
    next 0 is encountered */
71 2        call crlf;
72 2        call printx(a);

73 2    end print;

74 1    get$version: procedure byte;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/* returns current cp/m version */
75 2      return mon3(12,0);
76 2      end get$version;

77 1      select: procedure(d);
78 2      declare d byte;

79 2      call mon1(14,d);
80 2      end select;

81 1      check$user: procedure;
82 2      do forever;
83 3          if anything then return;
85 3          if dcnt = 0fff then return;
87 3          if dirbuf(ror(dcnt,3) and 110$0000b) = user$code then return;

89 3          dcnt = mon2(18,0);

90 3      end;
91 2      end check$user;

92 1      search: procedure(fcb);
93 2      declare fcb address;
94 2      declare fcb0 based fcb byte;

95 2      anything = (fcb0 = '?');
96 2      dcnt = mon2(17,fcb);
97 2      call check$user;
98 2      end search;

99 1      searchn: procedure;
100 2      dcnt = mon2(18,0);
101 2      call check$user;
102 2      end searchn;

103 1      cselect: procedure byte;

104 2      return mon2(25,0);
105 2      end cselect;

106 1      setdma: procedure(dma);
107 2      declare dma address;

108 2      call mon1(26,dma);
109 2      end setdma;

110 1      getalloca: procedure address;

111 2      return mon3(27,0);
112 2      end getalloca;

113 1      getlogin: procedure address;

114 2      return mon3(24,0);
115 2      end getlogin;

116 1      getpage: procedure byte;

```

/* return current disk number */

/* get base address of alloc vector */

/* get the login vector */

/* get the console page length */

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

117 2      SCBPB.where = 01ch;
118 2      return mon2(49,.SCBPB);

119 2      end getpage;

120 1      getpagemode: procedure byte;

121 2      SCBPB.where = 02ch;
122 2      return mon2(49,.SCBPB);

123 2      end getpagemode;

124 1      getNB: procedure byte;
125 2      SCBPB.where = 05dh;
126 2      return high(mon3(49,.SCBPB));
127 2      end getNB;

128 1      getrodisk: procedure address;

129 2      return mon3(29,0);
130 2      end getrodisk;

/*setind: procedure;
   call mon1(30,fcba);
end setind;
*/

131 1      set$dpb: procedure;

132 2      dpba = mon3(31,0);
133 2      end set$dpb;

134 1      getuser: procedure byte;

135 2      return mon2(32,0ffh);
136 2      end getuser;

/*setuser: procedure(user);
   declare user byte;

   call mon1(32,user);
end setuser;
*/

137 1      getfreesp: procedure(d);
138 2      declare d byte;

139 2      call mon1(46,d);
140 2      end getfreesp;

141 1      getlbl: procedure(d) byte;
142 2      declare d byte;

143 2      return mon2(101,d);
144 2      end getlbl;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

/* get the read-only disk vector */

/* set disk parameter block values */
/* base of dpb */

/* return current user number */

```

145 1  e$print: procedure(msg);
146 2      declare msg      address;

147 2      call print(.ERRORM);
148 2      call printx(msg);

149 2  end e$print;

```

```

/*****

```

Time & Date ASCII Conversion Code

```

*****/

```

```

150 1  declare tod$adr address;
151 1  declare tod based tod$adr structure (
      opcode byte,
      date address,
      hrs byte,
      min byte,
      sec byte,
      ASCII (21) byte );

152 1  declare string$adr address;
153 1  declare string based string$adr (1) byte;
154 1  declare index byte;

155 1  emitchar: procedure(c);
156 2      declare c byte;
157 2      string(index := index + 1) = c;
158 2      end emitchar;

159 1  emitn: procedure(a);
160 2      declare a address;
161 2      declare c based a byte;
162 2      do while c < '$';
163 3          string(index := index + 1) = c;
164 3          a = a + 1;
165 3      end;
166 2      end emitn;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

167 1  emit$bcd: procedure(b);
168 2      declare b byte;
169 2      call emitchar('0'+b);
170 2      end emit$bcd;

171 1  emit$bcd$pair: procedure(b);
172 2      declare b byte;
173 2      call emit$bcd(shr(b,4));
174 2      call emit$bcd(b and 0fh);
175 2      end emit$bcd$pair;

176 1  emit$colon: procedure(b);
177 2      declare b byte;
178 2      call emit$bcd$pair(b);

```

```

179 2      call emitchar('');
180 2      end emit$colon;

181 1      emit$bin$pair: procedure(b);
182 2      declare b byte;
183 2      call emit$bcd(b/10);      /* makes garbage if not < 10 */
184 2      call emit$bcd(b mod 10);
185 2      end emit$bin$pair;

186 1      emit$slant: procedure(b);
187 2      declare b byte;
188 2      call emit$bin$pair(b);
189 2      call emitchar('');
190 2      end emit$slant;

191 1      declar= chr byte;

192 1      gnc: procedure;
      /* get next command byte */
193 2      if chr = 0 then return;
195 2      if index = 20 then
196 2      do;
197 3          chr = 0;
198 3          return;
199 3      end;
200 2      chr = string(index := index + 1);
201 2      end gnc;

202 1      deblank: procedure;
203 2      do while chr = ' ';
204 3          call gnc;
205 3      end;
206 2      end deblank;

207 1      numeric: procedure byte;
      /* test for numeric */
208 2      return (chr - '0') < 10;
209 2      end numeric;

210 1      scan$numeric: procedure(lb,ub) byte;
211 2      declare (lb,ub) byte;
212 2      declare b byte;
213 2      b = 0;
214 2      call deblank;
215 2      if not numeric then call terminate;
217 2      do while numeric;
218 3          if (b and 1110$0000b) <> 0 then call terminate;
220 3          b = shl(b,3) + shl(b,1); /* b = b * 10 */
221 3          if carry then call terminate;
223 3          b = b + (chr - '0');
224 3          if carry then call terminate;
226 3          call gnc;
227 3          end;
228 2      if (b < lb) or (b > ub) then call terminate;
230 2      return b;
231 2      end scan$numeric;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

232 1  scan$delimiter: procedure(d,lb,ub) byte;
233 2      declare (d,lb,ub) byte;
234 2      call deblank;
235 2      if chr ◇ d then call terminate;
237 2      call gnc;
238 2      return scan$numeric(lb,ub);
239 2      end scan$delimiter;

240 1  declare
      base$year lit '78', /* base year for computations */
      base$day lit '0', /* starting day for base$year 0..6 */
      month$size (*) byte data
      /* jan feb mar apr may jun jul aug sep oct nov dec */
      ( 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31),
      month$days (*) address data
      /* jan feb mar apr may jun jul aug sep oct nov dec */
      ( 000,031,059,090,120,151,181,212,243,273,304,334);

241 1  leap$days: procedure(y,m) byte;
242 2      declare (y,m) byte;
      /* compute days accumulated by leap years */
243 2      declare yp byte;
244 2      yp = shr(y,2); /* yp = y/4 */
245 2      if (y and 11b) = 0 and month$days(m) < 59 then
          /* y not 00, y mod 4 = 0, before march, so not leap yr */
246 2          return yp - 1;
      /* otherwise, yp is the number of accumulated leap days */
247 2      return yp;
248 2      end leap$days;

249 1  declare word$value address;

250 1  bcd:
      procedure (val) byte;
251 2      declare val byte;
252 2      return shl((val/10),4) + val mod 10;
253 2      end bcd;

254 1  declare (month, day, year, hrs, min, sec) byte;

255 1  set$date$time: procedure;
256 2      declare
          (i, leap$flag) byte; /* temporaries */
257 2      month = scan$numeric(1,12) - 1;
      /* may be feb 29 */
258 2      if (leap$flag := month = 1) then i = 29;
260 2          else i = month$size(month);
261 2      day = scan$delimiter('/',1,i);
262 2      year = scan$delimiter('/',base$year,99);
      /* ensure that feb 29 is in a leap year */
263 2      if leap$flag and day = 29 and (year and 11b) ◇ 0 then
264 2          /* feb 29 of non-leap year */ call terminate;
      /* compute total days */
265 2      tod.date = month$days(month)
          + 365 * (year - base$year)
          + day
          - leap$days(base$year,0)

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

        + leap$days(year,month);

266 2      tod.hrs = bcd (scan$numeric(0,23));
267 2      tod.min = bcd (scan$delimiter(':',0,59));
268 2      if tod.opcode = 2 then
/* date, hours and minutes only */
269 2      do;
270 3          if chr = ':'
                then i = scan$delimiter(':',0,59);
272 3          tod.sec = 0;
273 3      end;
/* include seconds */
274 2      else tod.sec = bcd (scan$delimiter(':',0,59));

275 2      end set$date$time;

276 1      bcd$pair: procedure(a,b) byte;
277 2          declare (a,b) byte;
278 2          return shl(a,4) or b;
279 2      end bcd$pair;

280 1      compute$year: procedure;
/* compute year from number of days in word$value */
281 2      declare year$length address;
282 2      year = base$year;
283 2      do forever;
284 3          year$length = 365;
285 3          if (year and 11b) = 0 then /* leap year */
286 3              year$length = 366;
287 3          if word$value <= year$length then
288 3              return;
289 3          word$value = word$value - year$length;
290 3          year = year + 1;
291 3          end;
292 2      end compute$year;

293 1      declare
week$day byte, /* day of week 0 ... 6 */
day$list (*) byte data
('Sun$Mon$Tue$Wed$Thu$Fri$Sat$'),
leap$bias byte; /* bias for feb 29 */

294 1      compute$month: procedure;
295 2          month = 12;
296 2          do while month > 0;
297 3              if (month := month - 1) < 2 then /* jan or feb */
298 3                  leap$bias = 0;
299 3              if month$days(month) + leap$bias < word$value then return;
301 3              end;
302 2          end compute$month;

303 1      get$date$time: procedure;
/* get date and time */
304 2          hrs = tod.hrs;
305 2          min = tod.min;
306 2          sec = tod.sec;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

307 2      word$value = tod.date;
          /* word$value contains total number of days */
308 2      week$day = (word$value + base$day - 1) mod 7;
309 2      call compute$year;
          /* year has been set, word$value is remainder */
310 2      leap$bias = 0;
311 2      if (year and 11b) = 0 and word$value > 59 then
312 2          /* after feb 29 on leap year */ leap$bias = 1;
313 2      call compute$month;
314 2      day = word$value - (month$days(month) + leap$bias);
315 2      month = month + 1;
316 2      end get$date$time;

```

```

317 1      emit$date$time: procedure;
318 2          if tod.opcode = 0 then
319 2              do;
320 3              call emitn(.day$list(shl(week$day,2)));
321 3              call emitchar(' ');
322 3              end;
323 2          call emit$slant(month);
324 2          call emit$slant(day);
325 2          call emit$bin$pair(year);
326 2          call emitchar(' ');
327 2          call emit$colon(hrs);
328 2          call emit$colon(min);
329 2          if tod.opcode = 0 then
330 2              call emit$bcd$pair(sec);
331 2          end emit$date$time;

```

```

332 1      tod$ASCII:
          procedure (parameter);
333 2          declare parameter address;
334 2          declare ret address;

335 2          ret = 0;
336 2          tod$adr = parameter;
337 2          string$adr = .tod.ASCII;
338 2          if (tod.opcode = 0) or
              (tod.opcode = 3) then
339 2              do;
340 3              call get$date$time;
341 3              index = -1;
342 3              call emit$date$time;
343 3              end;
              else
344 2              do;
345 3              if (tod.opcode = 1) or
                  (tod.opcode = 2) then
346 3                  do;
347 4                  chr = string(index:=0);
348 4                  call set$date$time;
349 4                  ret = .string(index);
350 4                  end;
              else
351 3              do;
352 4              call terminate;
353 4              end;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

354 3      end;
355 2      end tod$ASCII;

```

```

/*****

```

TOD INTERFACE TO SHOW

```

*****/

```

```

356 1      declare lcltod structure (
           opcode byte,
           date address,
           hrs byte,
           min byte,
           sec byte,
           ASCII (21) byte );

```

```

/* declare extrnl$todadr address;
   declare extrnl$tod based extrnl$todadr structure (
       date address,
       hrs byte,
       min byte,
       sec byte );
*/

```

```

357 1      declare ret address;

```

```

/* display$tod;
   procedure;
       lcltod.opcode = 0;
       call move (5,.extrnl$tod.date,.lcltod.date);
       call tod$ASCII (.lcltod);
       call write$console (0dh);
       do i = 0 to 20;
           call write$console (lcltod.ASCII(i));
       end;
   end display$tod; */

```

```

358 1      display$ts;
           procedure (tsadr);
359 2          dcl i byte;
360 2          dcl tsadr address;
361 2          lcltod.opcode = 3;          /* display time and date stamp, no seconds */
362 2          call move (4,tsadr,.lcltod.date); /* don't copy seconds */
363 2          call tod$ASCII (.lcltod);
364 2          do i = 0 to 13;
365 3              call printchar (lcltod.ASCII(i));
366 3          end;
367 2      end display$ts;

```

```

/***** End TOD Code *****/

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* ****

*** BASIC ROUTINES ***

```

368 1  declare
      fcbmax literally '512'; /* max fcb count */

369 1  declare bpb address; /* bytes per block */

370 1  set$bpb: procedure;

371 2      call set$dpb;
372 2      bpb = shl(double(1),blkshf) * sectorlen;

373 2  end set$bpb;

374 1  select$disk: procedure(d);
375 2      declare d byte;

376 2      call select(cdisk:=d);
377 2      call set$bpb;

378 2  end select$disk;

379 1  getalloc: procedure(i) byte; /* return the ith bit of the alloc vector */
380 2      declare i address;

381 2      return
382 2      rol(alloc(shr(i,3)), (i and 111b) + 1);
      end getalloc;

/* ----- */

/* fill string @ s for c bytes with f */
383 1  fill: proc(s,f,c);
384 2      dcl s addr,
          (f,c) byte,
          a based s byte;

385 2      do while (c:=c-1)>255;
386 3          a = f;
387 3          s = stl;
388 3          end;
389 2      end fill;

```

COPYRIGHT © 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```
/* **** */
```

```
*** PRINT A NUMBER ***
```

```
**** */
```

```

390 1 declare
    val (7) byte initial(0,0,0,0,0,0,0), /* BCD digits */
    fac (7) byte initial(0,0,0,0,0,0,0), /* hi byte factor */
    f0 (7) byte initial(6,3,5,5,6,0,0), /* 65,536 */
    f1 (7) byte initial(2,7,0,1,3,1,0), /* 131,072 */
    f2 (7) byte initial(4,4,1,2,6,2,0), /* 262,144 */
    f3 (7) byte initial(8,8,2,4,2,5,0), /* 524,288 */
    f4 (7) byte initial(6,7,5,8,4,0,1), /* 1,048,576 */
    f5 (7) byte initial(2,5,1,7,9,0,2), /* 2,097,152 */
    f6 (7) byte initial(4,0,3,4,9,1,4), /* 4,194,304 */
    ptr (7) address initial(.f0,.f1,.f2,.f3,.f4,.f5,.f6);

                                /* print decimal value of address v */
391 1 pdecimal; procedure(v,prec,zerosup);
    /* print value v with precision prec (1,10,100,1000,10000)
    with leading zero suppression if zerosup = true */
392 2 declare
    v address, /* value to print */
    prec address, /* precision */
    zerosup byte, /* zero suppression flag */
    d byte; /* current decimal digit */

393 2 do while prec < 0;
394 3     d = v / prec; /* get next digit */
395 3     v = v mod prec; /* get remainder back to v */
396 3     prec = prec/10; /* ready for next digit */

397 3     if prec = 0 then go to pd0;
399 3     if d < 0 then go to pd0;
401 3     if zerosup then do;
403 4         call printb;
404 4         go to pd1;
405 4     end;
406 3     pd0: zerosup = false;
407 3         call printchar('0'+d);
408 3     pd1: end;

409 2 end pdecimal;

/* ----- */

                                /* BCD - convert 16 bit binary to
                                7 one byte BCD digits */
410 1 getbcd; procedure(value);
411 2 declare

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

        (value,prec) address,
        i byte;

412  2      prec = 10000;
413  2      i = 5;                      /* digits: 4,3,2,1,0 */
414  2      do while prec < 0;
415  3          val(i:=i-1) = value / prec; /* get next digit */
416  3          value = value mod prec;     /* remainder in value */
417  3          prec = prec / 10;
418  3          end;
419  2      end getbcd;

/* ----- */

                                /* print BCD number in val array */
420  1      printbcd: procedure;
421  2          declare
                (zerosup, i) byte;

422  2      pchar: procedure(c);
423  3          declare c byte;
424  3          if val(i) = 0 then
425  3              if zerosup then
426  3                  if i < 0 then do;
428  4                      call printb;
429  4                      return;
430  4                      end;
                /* else */
431  3          call printchar(c);
432  3          zerosup = false;
433  3      end pchar;

434  2      zerosup = true;
435  2      i = 7;
436  2      do while (i:=i-1) < -1;
437  3          call pchar('0'+val(i));
438  3          if i = 6 or i = 3 then
439  3              call pchar(',');
440  3          end;
441  2      end printbcd;

/* ----- */

                                /* add two BCD numbers result in second */
442  1      add: procedure(ap,bp);
443  2          declare
                (ap,bp)      address,
                a based ap (.) byte,
                b based bp (7) byte,
                (c,i)        byte;

444  2          c = 0;                      /* carry */
445  2          do i = 0 to 6;              /* 0 = LSB */
446  3              b(i) = a(i) + b(i) + c;
447  3              c = b(i) / 10;
448  3              b(i) = b(i) mod 10;
449  3          end;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

450 2      end add;
      /* ----- */

      /* print 3 byte value based at byte3adr */
451 1      p3byte: procedure(byte3adr);
452 2      declare
          i      byte,
          high$byte byte,
          byte3adr address,
          b3 based byte3adr structure (
              lword address,
              hbyte byte);

453 2      call fill(.val,0,7);
454 2      call fill(.fac,0,7);
455 2      call getbcd(b3.lword);      /* put 16 bit value in val */
456 2      high$byte = b3.hbyte;
457 2      do i = 0 to 6;              /* factor for bit i */
458 3      if high$byte then          /* LSB is 1 */
459 3      call add(ptr(i),.fac);      /* add in factor */
460 3      high$byte = shr(high$byte,1); /* get next bit */
461 3      end;
462 2      call add(.fac,.val);        /* add factor to value */
463 2      call printbcd;              /* print value */
464 2      end p3byte;

      /* divide 3 byte value by 8 */
465 1      shr3byte: procedure(byte3adr);
466 2      dcl byte3adr address,
          b3 based byte3adr structure (
              lword address,
              hbyte byte),
          temp1 based byte3adr (2) byte,
          temp2 byte;

467 2      temp2 = ror(b3.hbyte,3) and 11100000b; /* get 3 bits */
468 2      b3.hbyte = shr(b3.hbyte,3);
469 2      b3.lword = shr(b3.lword,3);
470 2      temp1(1) = temp1(1) or temp2; /* or in 3 bits from hbyte */
471 2      end shr3byte;

      /* multiply 3 byte value by #records per block */
472 1      shl3byte: procedure(byte3adr);
473 2      dcl byte3adr address,
          b3 based byte3adr structure (
              lword address,
              hbyte byte),
          temp1 based byte3adr (2) byte;

474 2      b3.hbyte = (rol(temp1(1),blkshf) and blkmask) or shl(b3.hbyte,blkshf);
475 2      b3.lword = shl(b3.lword,blkshf);
476 2      end shl3byte;

```

COPYRIGHT © 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

477 1  show$drive: procedure;
478 2      call printchar(cdisk+'A');
479 2      call printx(.,(' ',0));
480 2  end show$drive;

```

```

/* ****

```

```

*** CALCULATE SIZE ***

```

```

*****/

```

```

481 1  add$block: procedure(ak,ab);
482 2      declare (ak, ab) address;
          /* add one block to the kilobyte accumulator */
483 2      declare kaccum based ak address; /* kilobyte accum */
484 2      declare baccum based ab address; /* byte accum */
485 2      baccum = baccum + bpb;
486 2      do while baccum >= 1024;
487 3          baccum = baccum - 1024;
488 3          kaccum = kaccum + 1;
489 3      end;
490 2  end add$block;

```

```

491 1  count: procedure(mode) address;
492 2      declare mode byte; /* true if counting 0's */
          /* count kb remaining, kaccum set upon exit */
493 2      declare
          ka address, /* kb accumulator */
          ba address, /* byte accumulator */
          i address, /* local index */
          bit byte; /* always 1 if mode = false */
494 2      ka, ba = 0;
495 2      bit = 0;
496 2      do i = 0 to maxall;
497 3          if mode then bit = getalloc(i);
499 3          if not bit then call add$block(.ka,.ba);
501 3      end;
502 2      return ka;
503 2  end count;

```

```

/* ****

```

```

*** STATUS ROUTINES ***

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* characteristics of current drive */

```
504 1  drivestatus; procedure;
505 2      dcl b3a address,
          b3 based b3a structure (
              lword address,
              hbyte byte),

          psize address;
```

/* print 3 byte value */

```
506 2      pv3; procedure;
507 3          call crlf;
508 3          call p3byte(.dirbuf);
509 3          call printchar(' ');
510 3          call printb;
511 3      end pv3;
```

/* print address value v */

```
512 2      pv; procedure(v);
513 3          dcl v address;
514 3          b3.hbyte = 0;
515 3          b3.lword = v;
516 3          call pv3;
517 3      end pv;
```

/* print the characteristics of the currently selected drive */

```
518 2      b3a = .dirbuf;
519 2      call print(,(' ',0));
520 2      call show$drive;
521 2      call printx(,('Drive Characteristics',0));
522 2      b3.hbyte = 0;
523 2      b3.lword = maxall + 1;      /* = # blocks */
524 2      call shl3byte(.dirbuf);    /* # blocks * records/block */
525 2      call pv3;
526 2      call printx(,('128 Byte Record Capacity',0));
527 2      call shr3byte(.dirbuf);    /* divide by 8 */
528 2      call pv3;
529 2      call printx(,('Kilobyte Drive Capacity',0));
530 2      call pv(dirmax+1);
531 2      call printx(,('32 Byte Directory Entries',0));
532 2      call pv(shl(chksiz,2));
533 2      call printx(,('Checked Directory Entries',0));
534 2      call pv((extmsk+1) * 128);
535 2      call printx(,('Records / Directory Entry',0));
536 2      call pv(shl(double(1),blkshf));
537 2      call printx(,('Records / Block',0));
538 2      call pv(scptrk);
539 2      call printx(,('Sectors / Track',0));
540 2      call pv(offset);
541 2      call printx(,('Reserved Tracks',0));
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

542 2      psize = 128;          /* 2**psh * 128 */
543 2      if physhf > 0 then psize = shl(psize,physhf);

545 2      call pv(psize);
546 2      call printx(,('Bytes / Physical Record',0));
547 2      call crlf;

548 2      end drivestatus;
/* ----- */

/* ***** */

*** DISK STATUS ***

***** */

549 1      pvalue: procedure(v);
550 2      declare (d,zero) byte,
551 2      (k,v) address;
552 2      k = 10000;
553 2      zero = false;
554 2      do while k > 0;
555 3      d = low(v/k); v = v mod k;
556 3      k = k / 10;
557 3      if zero or k = 0 or d < 0 then
558 3      do; zero = true; call printchar('0'+d);
559 4      end;
560 3      end;
561 2      end pvalue;

564 1      prcount: procedure;

565 2      /* print the actual byte count */
566 3      if cversion < apm then do;
567 3      alloca = getalloca;
568 3      call pvalue(count(true));
569 3      end;
570 2      else do;
571 3      call setdma(.dirbuf);
572 3      call getfreesp(cdisk);
573 3      call shr3byte(.dirbuf);
574 3      call p3byte(.dirbuf);
575 3      end;
576 2      call printchar('K');
577 2      end prcount;

578 1      stat: procedure(rodisk);
579 2      declare rodisk address;

580 2      call crlf;
581 2      call show$drive;
582 2      call printchar('R');
```

CP/M - PLUS V 3.0

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # CP3-135

```

583 2      if low(rodisk) then
584 2          call printchar('0'); else
585 2          call printchar('W');
586 2      call printx(.,', Space: ',0));
587 2      call prcount;
588 2      end stat;

589 1  prstatus: procedure;          /* print the status of the disk system */
590 2      declare (login, rodisk) address;
591 2      declare (d,save) byte;

592 2          if once$only then return;          /* only execute this once if
                                                all was specified > 1 */

594 2      save = cdisk;
595 2      login = getlogin;          /* login vector set */
596 2      rodisk = getrodisk;        /* read only disk vector set */

597 2      d = 0;
598 2      do while login <> 0;
599 3          if low(login) then do;
601 4              if not all then do;          /* do specified disk */
603 5                  if d = save then call stat(rodisk);
605 5              end;

606 4              else do;
607 5                  call select$disk(d);      /* do all disks */
608 5                  call stat(rodisk);
609 5              end;
610 4          end;

611 3          login = shr(login,1); rodisk = shr(rodisk,1);
613 3          d = d + 1;
614 3      end;

615 2      if all then once$only = true;
617 2      call crlf;

618 2      end prstatus;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

*** USER STATUS ***

```

619 1  prdir: procedure;

620 2      call crlf;
621 2      call crlf;
622 2      call show$drive;

```

```

623 2      if nSFCB > 0 then do;
625 3          call printx(,('Number of time/date directory entries: ',0));
626 3          call pdecimal(nSFCB,1000,true);
627 3          call crlf;
628 3          call show$drive;
629 3      end;

630 2      call printx(,('Number of free directory entries:      ',0));
631 2      call pdecimal(free$dir,1000,true);
632 2      call crlf;

633 2      end prdir;

634 1      get$usr$files: procedure;
635 2          declare ufcB(*) byte data ('????????????',0,0,0),
              (i,j) byte,
              nfcbs address,
              extptr address,
              modptr address,
              fmod based modptr byte,
              fext based extptr byte;

636 2          do i = 0 to 15;
637 3              user(i),used(i) = 0;
638 3          end;
639 2          nSFCB = 0;

640 2          call setdma(.dirbuf);
641 2          call search(.ufcB);
642 2          nfcbs = 0;

643 2          do while dcnt <> 255;
644 3              j = shl(dcnt,5);          /* which fcb in dirbuf */

645 3      ge0:          if (i := dirbuf(j)) <> 0e5h then do;
647 4                  if i <> 33 then do;          /* SFCB ? */
649 5                      extptr = .dirbuf(j + 12);
650 5                      modptr = extptr + 2;
651 5                      nfcbs = nfcbs + 1;
652 5                      j = i;          /* save for xfcB test */
653 5                      user(i := i and 0fh) = true;

654 5                      if j > 15 then go to ge2;
656 5                      if fext > extask then go to ge2;
658 5                      if fmod = 0 then used(i) = used(i) + 1;
660 5                      end;
661 4                      else nSFCB = nSFCB + 1;
662 4                      end;

663 3      ge2:          call searchn;
664 3      end;

665 2      done$drive(cdisk) = true;
666 2      if nSFCB > 0 then nSFCB = shr(dirmx+1,2);          /* because search ends
                                                             at high water mark*/

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

668 2      free$dir = ((dirmax + 1) - nSFCB) - nfcbs;

669 2      end get$usr$files;

670 1      userstatus: procedure;          /* display active user numbers */
671 2      declare i byte;

672 2      call crlf;
673 2      call show$drive;
674 2      call printx(,('Active User : ',0));
675 2      call pdecimal(getuser,1000,true);
676 2      call crlf;
677 2      call show$drive;
678 2      call printx(,('Active Files: ',0));

679 2      if not done$drive(cdisk) then call get$usr$files;

681 2      do i = 0 to last(user);
682 3          if user(i) then call pdecimal(i,1000,true);
684 3      end;

685 2      call crlf;
686 2      call show$drive;
687 2      call printx(,('# of files : ',0));
688 2      do i = 0 to last(user);
689 3          if user(i) then call pdecimal(used(i),1000,true);
691 3      end;

692 2      call prdir;

693 2      end userstatus;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* ***** */

*** DISK & FILE STATUS ***

***** */

```

694 1      directory: procedure;

695 2      if not done$drive(cdisk) then call get$usr$files;
697 2      call prdir;

698 2      end directory;

```

L A B E L S T A T U S

```

699 1  readlbl: proc(relog);
700 2      declare relog  byte,
           d             byte data('?');

701 2      call setdma(.dirbuf);
702 2      call search(.d);
703 2      if relog > 0 then return;

705 2      do while dcnt <> 0ffh;
706 3          if dirbuf(ror(dcnt,3) and 110$0000b)=20H then return;
708 3          call searchn;
709 3      end;

710 2  end readlbl;

/* HEADER */

711 1  dcl label1 (*) byte data (
       'Directory  Passwds Stamp  Stamp',0);
712 1  dcl label2 (*) byte data (
       'Label      Req'd      ',0);
713 1  dcl label3 (*) byte data (
       ' Update Label Created Label Updated',0)

;
714 1  dcl label4 (*) byte data (
       '-----',0

);

715 1  labelstatus: procedure;
716 2      dcl lbl          byte;
717 2      dcl fnam lit '11';
718 2      dcl ftyp lit '9';
719 2      dcl fcbp address;
720 2      dcl fcbv based fcbp (32) byte; /* template over dirbuf */

721 2      printfn: proc; /* print file name */
722 3          declare k byte;

723 3          do k = 1 to fnam;
724 4              if k = ftyp then
725 4                  call printchar('.');
726 4                  call printchar(fcbv(k) and 7fh);
727 4              end;
728 3      end printfn,

729 2      lbl = getlbl(cdisk);
730 2      if lbl > 0 then do;
732 3          call readlbl(0);
733 3          fcbp = shl(dcnt,5) + .dirbuf;

734 3          call print(.(('Label for drive ',0))); /* print heading */
735 3          call show$drive;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 879
 PACIFIC GROVE, CA 93950

SER. # _____

```

736 3      call crlf;
737 3      call print(.label1);
738 3      call print(.label2);
739 3      if (lbl and 40h) = 40h then
740 3          call printx(,('Access',0));
      else
741 3          call printx(,('Create',0));
742 3      call printx(.label3);
743 3      call print(.label4);
744 3      call crlf;
745 3      call printfn;
746 3      if not NONBANK and ((lbl and 80h) = 80h) then
747 3          call printx(,(' on ',0));
      else
748 3          call printx(,(' off ',0));

749 3      if (lbl and 40h) = 40h then
750 3          call printx(,(' on ',0));
751 3      else if (lbl and 10h) = 10h then
752 3          call printx(,(' on ',0));
753 3      else call printx(,(' off ',0));

754 3      if (lbl and 20h) = 20h then
755 3          call printx(,(' on ',0));
      else
756 3          call printx(,(' off ',0));

757 3      call printx(,(' ',0));
758 3      call display$ts(.fcbv(24));
759 3      call printx(,(' ',0));
760 3      call display$ts(.fcbv(28));
761 3      end;
762 2      else do;
763 3          call e$print(.err$noLabel);
764 3          call printchar(cdisk+'A');
765 3      end;

766 2      call crlf;

767 2      end labelstatus;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

$reject
/* **** */

```

```

* * * Option scanner * * *

```

```

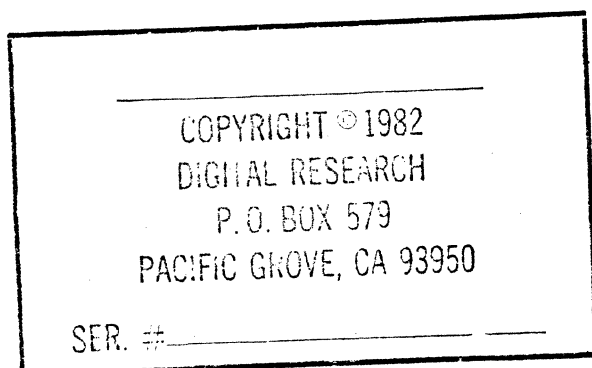
* * * * *

```

```

768 1  separator: procedure(character) byte;
                                     /* determines if character is a
                                     delimiter and which one */
769 2      declare k      byte,
               character  byte;
770 2      k = 1;
771 2  loop:  if delimiters(k) = end$list then return(0);
773 2      if delimiters(k) = character then return(k); /* null = 25 */
775 2      k = k + 1;
776 2      go to loop;
777 2  end separator;
778 1  opt$scanner: procedure(list$ptr, off$ptr) byte;
                                     /* scans the list pointed at by idxptr
                                     for any strings that are in the
                                     list pointed at by list$ptr.
                                     Offptr points at an array that
                                     contains the indices for the known
                                     list. Idxptr points at the index
                                     into the list. If the input string
                                     is unrecognizable then the index is
                                     0, otherwise > 0.

```



First, find the string in the known list that starts with the same first character. Compare up until the next delimiter on the input. if every input character matches then check for uniqueness. Otherwise try to find another known string that has its first character match, and repeat. If none can be found then return invalid.

To test for uniqueness, start at the next string in the known list and try to get another match with the input. If there is a match then return invalid.

else move pointer past delimiter and return.

```

P.Balma */

```

```

779 2  declare

```

```

        buff      based buf$ptr (1) byte,
        off$ptr    address,
        list$ptr   address;

780  2      declare
            i      byte,
            j      byte,
            list    based list$ptr (1) byte,
            offsets based off$ptr (1) byte,
            wrd$pos byte,
            character byte,
            letter$in$word byte,
            found$first byte,
            start   byte,
            index   byte,
            save$index byte,
            (len$new, len$found) byte,
            valid   byte;

        /*****
        /*          internal subroutines          */
        *****/

781  2      check$in$list: procedure;

                                /* find known string that has a match with
                                input on the first character. Set index
                                = invalid if none found. */

782  3      declare i      byte;

783  3      i = start;
784  3      wrd$pos = offsets(i);
785  3      do while list(wrd$pos) <> end$list;
786  4          i = i + 1;
787  4          index = i;
788  4          if list(wrd$pos) = character then return;
790  4          wrd$pos = offsets(i);
791  4      end;

                                /* could not find character */
792  3      index = 0;
793  3      return;
794  3      end check$in$list;

795  2      setup: procedure;
796  3      character = buff(0);
797  3      call check$in$list;
798  3      letter$in$word = wrd$pos;

                                /* even though no match may have occurred, position
                                to next input character. */

799  3      i = 1;
800  3      character = buff(1);
801  3      end setup;

802  2      test$letter: procedure;

                                /* test each letter in input and known string */

803  3      letter$in$word = letter$in$word + 1;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/* too many chars input? 0 means
   past end of known string */
804 3      if list(letter$in$word) = end$of$string then valid = false;
           else
806 3      if list(letter$in$word) <> character then valid = false;

           i = i + 1;
809 3      character = buff(i);

810 3      end test$letter;

811 2      skip: procedure;

/* scan past the offending string;
   position buf$ptr to next string...
   skip entire offending string;
   ie., falseopt=mod, [note: comma or
   space is considered to be group
   delimiter] */

812 3      character = buff(i);
813 3      delimiter = separator(character);
814 3      do while ((delimiter <> 2) and (delimiter <> 4) and (delimiter <> 5)
                  and (delimiter <> 25));
815 4          i = i + 1;
816 4          character = buff(i);
817 4          delimiter = separator(character);
818 4      end;
819 3      endbuf = i;
820 3      buf$ptr = buf$ptr + endbuf + 1;
821 3      return;
822 3      end skip;

823 2      eat$blanks: procedure;

824 3      declare charac based buf$ptr byte;

825 3      do while(delimiter := separator(charac)) = SPACE;
826 4          bufptr = buf$ptr + 1;
827 4      end;

828 3      end eat$blanks;

/*****
/*      end of internals      */
*****/

/* start of procedure */

829 2      call eat$blanks;
830 2      start = 0;
831 2      call setup;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* match each character with the option
 for as many chars as input
 Please note that due to the array
 indices being relative to 0 and the

use of index both as a validity flag
and as a index into the option/modes
list, index is forced to be +1 as an
index into array and 0 as a flag*/

```

832 2      do while index <> 0;
833 3          start = index;
834 3          delimiter = separator(character);

/* check up to input delimiter */

835 3          valid = true;          /* test$letter resets this */
836 3          do while delimiter = 0;
837 4              call test$letter;
838 4              if not valid then go to exit1;
840 4              delimiter = separator(character);
841 4          end;

842 3          go to good;

/* input ~= this known string;
get next known string that
matches */

843 3      exit1:    call setup;
844 3              end;

/* fell through from above, did
not find a good match*/
/* skip over string & return*/

845 2      endbuf = i;
846 2      call skip;
847 2      return(index);

/* is it a unique match in options
list? */

848 2      good:    endbuf = i;
849 2              len$found = endbuf;
850 2              save$index = index;
851 2              valid = false;
852 2      next$opt:
853 2              start = index;
854 2              call setup;
855 2              if index = 0 then go to finished;

/* look at other options and check
uniqueness */

856 2      len$new = offsets(index + 1) - offsets(index) - 1;
857 2      if len$new = len$found then do;
859 3          valid = true;
860 3          do j = 1 to len$found;
861 4              call test$letter;
862 4              if not valid then go to next$opt;
864 4          end;
865 3      end;
866 2      else go to next$opt;

/* fell through...found another valid
match --> ambiguous reference */

867 2      index = 0;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

868 2      call skip;          /* skip input field to next delimiter*/
869 2      return(0);

870 2      finished:          /* unambiguous reference */
      index = save$index;
871 2      buf$ptr = buf$ptr + endbuf;
872 2      call eat$blanks;
873 2      if delimiter <> 0 then buf$ptr = buf$ptr + 1;
875 2      else delimiter = SPACE;
876 2      return(index);

877 2      end opt$scanner;

878 1      error$prt:  procedure;
879 2      declare i    byte,
      t      address,
      char   based t byte;

880 2      t = buf$ptr - endbuf - 1;
881 2      do i = 1 to endbuf;
882 3          call printchar(char);
883 3          t = t + 1;
884 3      end;

885 2      end error$prt;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

\$object

/* **** */

*** EXECUTE ***

/* **** */

```
886 1  do$option: procedure(i);
887 2      declare i      byte;

888 2      if opt$map(i).option(opt$space) < 0 then call prstatus;
890 2      if opt$map(i).option(opt$label) < 0 then call labelstatus;
892 2      if opt$map(i).option(opt$drive) < 0 then call drivestatus;
894 2      if opt$map(i).option(opt$user) < 0 then call userstatus;
896 2      if opt$map(i).option(opt$dir) < 0 then call directory;

898 2  end do$option;
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
$object
```

```
/* **** */
```

```
*** PARSING ***
```

```
*****/
```

```
899 1 declare character based buf$ptr byte;
900 1 setdef$drive: procedure;
901 2     if drive = Offh then do;
903 3         drive = cdisk;
904 3         drives(drive) = drive;
905 3     end;
906 2     return;
907 2 end setdef$drive;
908 1 parseoptions: procedure byte;
                                /* find all options within [...] */
909 2     buf$ptr = buf$ptr + 1;
910 2     delimiter = separator(character);
911 2     call setdef$drive;
912 2     if delimiter = 0 then go to preloop;
914 2     if delimiter < RBRACKET then
915 2         if delimiter < EOS then go to preloop;
                                /* [], turn on space */
917 2     opt$map(drive).option(opt$space) = 1;
918 2     buf$ptr = buf$ptr + 1;
919 2     return(2);
920 2 preloop:
921 2     if opt$map(drive).option(opt$space) = Offh then /* reset forced space*/
922 2         opt$map(drive).option(opt$space) = 0;
922 2 loop:  if (opt$index := optscanner(.options,.off$opt)) = 0 then go to error;
924 2     if opt$index = opt$page then PAGE = true;
926 2     else if opt$index = opt$nopage then PAGE = false;
928 2     else opt$map(drive).option(opt$index - 1) = 1;
929 2     go to looptest;
930 2 error: call e$print(.err$unrecopt);
931 2     call print(.eoption);
932 2     call error$prt;
933 2 looptest:
```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

```

        if delimiter = EOS then return(25);
935 2      if delimiter = RBRACKET then return(2);

937 2      go to loop;

938 2      end parseoptions;

939 1      parsedir: procedure;

940 2      declare dirindex      byte;

941 2      if (dir$index := optscanner(.dirs,.off$dirs)) = 0 then go to error1;

943 2      drive = dir$index - 1;
944 2      drives(drive) = drive;
945 2      opt$map(drive).option(opt$space) = Offh; /* only drive!, reset
                                                    if other options and
                                                    not space picked */

946 2      if delimiter <> COLON then buf$ptr = buf$ptr - 1;

948 2      return;

949 2      error1: call e$print(.err$unrecd);
950 2      dprint: call print(.dirdrive);
951 2      call error$prt;
952 2      call terminate;

953 2      end parsedir;

954 1      parser: procedure;

955 2      drive = Offh;

956 2      if (delimiter := separator(character)) = EOS then do;
958 3          call setdef$drive;
959 3          opt$map(drive).option(opt$space) = 1;          /* default*/
960 3          all = true;
961 3          return;
962 3      end;

963 2      loop: if delimiter = LBRACKET then delimiter = parseoptions;
965 2      else if delimiter = 0 then call parsedir;

967 2      else do;
968 3          if delimiter <> COMMA then
969 3          if delimiter <> SPACE then go to error;

971 3          drive = Offh;
972 3          buf$ptr = buf$ptr + 1;
973 3      end;

974 2      looptest:
          if delimiter <> EOS then
975 2          if (delimiter := separator(character)) <> EOS then go to loop;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
977 2      return;

978 2      error: call e$print(.err$input);
979 2          call print(.input);
980 2          call error$pri;
981 2          call terminate;

982 2      end parser;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

\$object

/*****

*** MAIN PROGRAM ***

*****/

```

983 1      declare
          i      byte initial(1);

984 1      plm:
          cversion = get$version;
985 1      if cversion < mp then call e$print(.err$version);
987 1      else do;

988 2          do while buff(i) = ' ';
989 3              i = i + 1;
990 3          end;
991 2          buf$ptr = .buff(i);

992 2          cdisk = cselect;
993 2          user$code = getuser;

994 2          do i = 0 to 15;
995 3              drives(i) = 0ffh;
996 3          end;

997 2          if getpagemode = 0 then PAGE = true;
999 2          line$page = getpage;
1000 2          line$out = 0;
1001 2          if getNB = 0 then NONBANK = true;

1003 2          call parser;

1004 2          do i = 0 to 15;
1005 3              if (drive := drives(i)) <> 0ffh then do;
1007 4                  call select$disk(drives(i));
1008 4                  call readbl(1); /* force login
                                   by wild card drive
                                   search. */

1009 4                  call do$option(i);
1010 4              end;
1011 3          end;

1012 2      end;
1013 1      call terminate;

1014 1      end;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 679

PACIFIC GROVE, CA 93950

SER. # _____

MODULE INFORMATION:

CODE AREA SIZE = 1CDFH 7391D

VARIABLE AREA SIZE = 026CH 620D

PL/M-80 COMPILER - CP/M 3.0 --- SHOW 3.1

MAXIMUM STACK SIZE = 0016H 22D
1877 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

0000 SHOW				
0000 MCDIAA				
0004 BLISK	0080 BUFF	0080 BUFFA	0050 CHDRV	0000 CPU
007C CR	006D DOLLA	005C FCB	006C FCB16	005C FCBA
005C IFCB	005C IFCBA	0003 IOBYTE	0053 LENO	0056 LEN1
0006 MAXB	0006 MEMSIZ	0005 MON1	0005 MON2	0005 MON2A
0005 MON3	0000 OFFSET	006E PARMA	0051 PASS0	0054 PASS1
007F RO	007D RR	007D RRECA	007F RRECO	005C SFCB
0080 TBUFF				
0000 SHOW				
21E9 MEMORY	05E5 PLM	0180 COPYRIGHT	01A6 VERDATE	01AE VERSION
1F7D DCNT	1F7E ANYTHING	1F7F DIRBUF	1FFF LINEPAGE	2000 LINEOUT
2001 DRIVES	2011 DRIVE	2012 ALL	2013 ONCEONLY	2014 DONDRIIVE
2024 PAGE	2025 NONBANK	2026 USER	2036 USED	2056 FREEDIR
2058 NSFCB	205A SCBFB	01B6 ERRORM	01BE INPUT	01C6 EOPTION
01CF DIRDRIVE	01D7 ERRUNRECOPT		01EC ERRUNRECD	0200 ERRVERSION
021B ERRNOLABEL	023F ERRINPUT	0253 OPTDIR	0254 OPTDRIVE	0255 OPTLABEL
0256 OPTSPACE	0257 OPTUSER	0258 OPTPAGE	0259 OPTNOFAGE	025A DIRS
028A OPTIONS	02B9 OFFDIRS	02CA OFFOPT	02D2 ENDLIST	
02D3 ENDOFSTRING		02D4 DELIMITERS	02EF SPACE	02F0 EOS
02F1 COMMA	02F2 COLON	02F3 LBRACKET	02F4 RBRACKET	205E OPTMAP
20C7 J	20C8 BUFPTR	20CA OPTINDEX	20CB ENDBUF	20CC DELIMITER
20CD USERCODE	20CE CVERSION	20D0 CDISK	20D1 DPBA	20D3 ALLOCA
06C9 PRINTCHAR	20D5 CHAR	06D9 PRINTB	06DF PRINTX	20D6 A
0700 BREAK	0709 CRLF2	0714 TERMINATE	0720 CRLF	0769 PRINT
20D8 A	077B GETVERSION	0785 SELECT	20DA D	0795 CHECKUSER
07CC SEARCH	20DB FCB	07EE SEARCHN	07FD CSELECT	0806 SETDMA
20DD DMA	0816 GETALLOCA	081F GETLOGIN	0828 GETPAGE	
0836 GETPAGEMODE		0844 GETNB	0853 GETRODISK	085C SETDPB
0868 GETUSER	0871 GETFREESP	20DF D	0881 GETLBL	20E0 D
0891 EPRINT	20E1 MSG	20E3 TODADR	20E5 STRINGADR	20E7 INDEX
08A6 EMITCHAR	20E8 C	08BD EMITN	20E9 A	08EC EMITBCD
20EB B	08FA EMITBCDPAIR		20EC B	0915 EMITCOLON
20ED B	0926 EMITBINPAIR		20EE B	094B EMITSLANT
20EF B	20F0 CHR	095C GNC	0986 DEBLANK	0995 NUMERIC
099E SCANNUMERIC		20F1 LB	20F2 UB	20F3 B
0A19 SCANDELIMITER		20F4 D	20F5 LB	20F6 UB
02F5 MONTHSIZE	0301 MONTHDAYS	0A43 LEAPDAYS	20F7 Y	20F8 M
20F9 YP	20FA WORDVALUE	0A7F BCD	20FC VAL	20FD MONTH
20FE DAY	20FF YEAR	2100 HRS	2101 MIN	2102 SEC
0AA2 SETDATETIME		2103 I	2104 LEAPFLAG	0BC3 BCDPAIR
2105 A	2106 B	0BD5 COMPUTEYEAR		2107 YEARLENGTH
2109 WEEKDAY	0319 DAYLIST	210A LEAPBIAS	0C13 COMPUTEMONTH	
0C51 GETDATETIME		0CCF EMITDATETIME		0D27 TODASCII
210B PARAMETER	210D RET	210F LCLTOD	212A RET	0D9F DISPLAYTS
212C TSADR	212E I	212F BPB	0DEE SETBPB	0E07 SELECTDISK
2131 D	0E19 GETALLOC	2132 I	0E3C FILL	2134 S
2136 F	2137 C	2138 VAL	213F FAC	2146 F0
214D F1	2154 F2	215B F3	2162 F4	2169 F5
2170 F6	2177 PTR	0E67 PDECIMAL	2185 V	2187 PREC
2189 ZEROSUP	218A D	0ED2 PDO	0EE0 PD1	0EE4 GETBCD
218B VALUE	218D PREC	218F I	0F39 PRINTBCD	2190 ZEROSUP
2191 I	0F7E PCHAR	2192 C	0FB1 ADD	2193 AP
2195 BP	2197 C	2198 I	1031 P3BYTE	2199 BYTE3ADR
219B I	219C HIGHBYTE	10A6 SHR3BYTE	219D BYTE3ADR	219F TEMP2
10E9 SHL3BYTE	21A0 BYTE3ADR	1138 SHOWDRIVE	1148 ADDBLOCK	21A2 AK
21A4 AB	1193 COUNT	21A6 MODE	21A7 KA	21A9 BA
21AB I	21AD SIT	11F0 DRIVESTATUS		21AE B3A
21B0 FSIZE	12F5 PV3	1307 PV	21B2 V	1324 PVALUE
21B4 V	21B6 D	21B7 ZERO	21B8 K	139B PRCOUNT
13DB STAT	21BA RODISK	1408 PRSTATUS	21BC LOGIN	21BE RODISK
21C0 D	21C1 SAVE	1498 PRDIR	14DA GETUSRFILES	
0335 UFCB	21C2 I	21C3 J	21C4 NFCBS	21C6 EXTPTR

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

21C8 MODPTR	1538 GEO	15D4 GE2	1623 USERSTATUS	21CA I
16DC DIRECTORY	16F2 READLBL	21CB RELOG	0344 D	0345 LABEL1
036A LABEL2	0382 LABEL3	03AA LABEL4	1731 LABELSTATUS	
21CC LBL	21CD FCBP	1842 PRINTFN	21CF K	1879 SEPARATOR
21D0 CHARACTER	21D1 K	1882 LOOP	1884 OPTSCANNER	21D2 LISTPTR
21D4 OFFPTR	21D6 I	21D7 J	21D8 WRDPOS	21D9 CHARACTER
21DA LETTERINWORD		21DB FOUNDFIRST	21DC START	21DD INDEX
21DE SAVEINDEX	21DF LENNEW	21E0 LENFOUND	21E1 VALID	
19E4 CHECKINLIST		21E2 I	1A3F SETUP	1A5D TESTLETTER
1AA9 SKIP	1B29 EATBLANKS	190C EXIT1	191F GOOD	1936 NEXTOPT
19B1 FINISHED	1B45 ERRORPRT	21E3 I	21E4 T	1B7A DOOPTION
21E6 I	1C1F SETDEFDRIVE		1C3B PARSEOPTIONS	
1C94 PRELOOP	1CCA LOOP	1D1E ERROR	1D2D LOOPTEST	1D4B PARSEDIR
21E7 DIRINDEX	1D9E ERROR1	1DA4 DPRINT	1DB1 PARSE	1DE9 LOOP
1E2D LOOPTEST	1E4C ERROR	21E8 I		

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

0000 SHOW#
0000 MCD80A#
0000 SHOW#

06C9 24	06CD 26	06DB 27	06D9 28	06D9 29
06DE 30	06DF 31	06E5 34	06EE 35	06F5 36
06FC 37	06FF 38	0700 39	0700 40	0709 41
0709 42	0709 43	070E 44	0713 45	0714 46
0714 47	0717 48	071F 49	0720 50	0720 51
0727 53	072B 54	0737 56	073A 57	073D 58
0743 59	074A 60	074D 61	075A 62	075D 63
0762 64	0765 65	0765 66	0765 67	0768 68
0769 69	076F 71	0772 72	077A 73	077B 74
077B 75	0785 76	0785 77	0789 79	0794 80
0795 81	0795 82	0795 83	079C 84	079D 85
07A5 86	07A6 87	07BC 88	07BD 89	07C8 90
07CB 91	07CC 92	07D2 95	07DE 96	07EA 97
07ED 98	07EE 99	07EE 100	07F9 101	07FC 102
07FD 103	07FD 104	0806 105	0806 106	080C 108
0815 109	0816 110	0816 111	081F 112	081F 113
081F 114	082B 115	082B 116	082B 117	082D 118
0836 119	0836 120	0836 121	083B 122	0844 123
0844 124	0844 125	0849 126	0853 127	0853 128
0853 129	085C 130	085C 131	085C 132	0867 133
0868 134	0868 135	0871 136	0871 137	0875 139
0880 140	0881 141	0885 143	0891 144	0891 145
0897 147	089D 148	08A5 149	08A6 155	08AA 157
08BC 158	08BD 159	08C3 162	08CC 163	08E1 164
08EB 165	08EB 166	08EC 167	08F0 169	08F9 170
08FA 171	08FE 173	090B 174	0914 175	0915 176
0919 178	0920 179	0925 180	0926 181	092A 183
093A 184	094A 185	094B 186	094F 188	0956 189
095B 190	095C 192	095C 193	0964 194	0965 195
096D 197	0972 198	0973 199	0973 200	0985 201
0986 202	0986 203	098E 204	0991 205	0994 206
0995 207	0995 208	099E 209	099E 210	09A4 213
09A9 214	09AC 215	09B3 216	09B6 217	09BD 218
09C7 219	09CA 220	09DB 221	09E0 222	09E3 223
09ED 224	09F2 225	09F5 226	09FB 227	09FB 228
0A12 229	0A15 230	0A19 231	0A19 232	0A24 234
0A27 235	0A31 236	0A34 237	0A37 238	0A43 239
0A43 241	0A49 244	0A53 245	0A76 246	0A7B 247
0A7F 248	0A7F 250	0A83 252	0AA2 253	0AA2 255
0AA2 257	0AAD 258	0ABC 259	0AC4 260	0AD1 261
0AE0 262	0AED 263	0B0B 264	0B0E 265	0B59 266
0B6C 267	0B82 268	0B8B 270	0B93 271	0BA0 272
0BA9 273	0BAC 274	0BC2 275	0BC3 276	0BC9 278
0BD5 279	0BD5 280	0BD5 282	0BDA 283	0BDA 284
0BE0 285	0BEA 286	0BF0 287	0BFC 288	0BFD 289
0C0B 290	0C0F 291	0C12 292	0C13 294	0C13 295
0C18 296	0C21 297	0C2D 298	0C32 299	0C4C 300
0C4D 301	0C50 302	0C51 303	0C51 304	0C5C 305
0C65 306	0C6E 307	0C79 308	0CB6 309	0CB9 310
0C8E 311	0CA9 312	0CAE 313	0CB1 314	0CCC 315
0CCE 316	0CCF 317	0CCF 318	0CDB 320	0CE9 321
0CEE 322	0CEE 323	0CF5 324	0CFC 325	0D03 326
0D08 327	0D0F 328	0D16 329	0D1F 330	0D26 331
0D27 332	0D2D 335	0D33 336	0D39 337	0D40 338
0D57 340	0D5A 341	0D5F 342	0D62 343	0D65 345
0D7C 347	0D88 348	0D8B 349	0D98 350	0D9B 352
0D9E 353	0D9E 354	0D9E 355	0D9F 358	0DA5 361
0DAA 362	0DBE 363	0DC4 364	0DD2 365	0DE3 366
0DED 367	0DEE 370	0DEE 371	0DF1 372	0E06 373
0E07 374	0E0B 376	0E15 377	0E18 378	0E19 379

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

0E1F 381	0E3C 382	0E3C 383	0E49 385	0E55 386
0E5C 387	0E63 388	0E66 389	0E67 391	0E76 393
0E82 394	0E90 395	0E9D 396	0EAB 397	0EB7 398
0EBA 399	0EC2 400	0ECS 401	0ECC 403	0ECF 404
0ED2 405	0ED2 406	0ED7 407	0EE0 408	0EE3 409
0EE4 410	0EEA 412	0EF0 413	0EF5 414	0F01 415
0F1A 416	0F27 417	0F35 418	0F38 419	0F39 420
0F7E 422	0F82 424	0F91 425	0F98 426	0FA0 428
0FA3 429	0FA4 430	0FA4 431	0FAB 432	0FB0 433
0F39 434	0F3E 435	0F41 436	0F4D 437	0F5D 438
0F75 439	0F7A 440	0F7D 441	0FB1 442	0FRB 444
0FC0 445	0FCC 446	0FF1 447	1008 448	1026 449
1030 450	1031 451	1037 453	1042 454	104D 455
1056 456	105F 457	106D 458	1074 459	1087 460
108F 461	1099 462	10A2 463	10A5 464	10A6 465
10AC 467	10BA 468	10CB 469	10DB 470	10E8 471
10E9 472	10EF 474	1121 475	1137 476	1138 477
1138 478	1141 479	1147 480	1148 481	1152 485
1163 486	1170 487	1182 488	118F 489	1192 490
1193 491	1197 494	11A0 495	11A5 496	11BC 497
11C3 498	11CE 499	11D6 500	11DF 501	11EC 502
11F0 503	1F0 504	12F5 506	12F5 507	12F8 508
12FE 509	1303 510	1306 511	1307 512	130D 514
1314 515	1320 516	1323 517	11F0 518	11F6 519
11FC 520	11FF 521	1205 522	120C 523	121D 524
1223 525	1226 526	122C 527	1232 528	1235 529
123B 530	1249 531	124F 532	1261 533	1267 534
127E 535	1284 536	1295 537	129B 538	12A4 539
12AA 540	12B7 541	12BD 542	12C3 543	12D0 544
12E3 545	12EB 546	12F1 547	12F4 548	1324 549
132A 551	1330 552	1335 553	1341 554	134F 555
135C 556	136A 557	1389 559	138E 560	1397 561
1397 562	139A 563	139B 564	139B 565	13A6 567
13AC 568	13B6 569	13B9 571	13BF 572	13C6 573
13CC 574	13D2 575	13D2 576	13D7 577	13D8 578
13DE 580	13E1 581	13E4 582	13E9 583	13F1 584
13F9 585	13FE 586	1404 587	1407 588	1408 589
1408 592	140F 593	1410 594	1416 595	141C 596
1422 597	1427 598	1433 599	143B 601	1443 603
144D 604	1455 605	1458 607	145F 608	1467 609
1467 610	1467 611	1474 612	1481 613	1485 614
1488 615	148F 616	1494 617	1497 618	1498 619
1498 620	149B 621	149E 622	14A1 623	14AC 625
14B2 626	14BE 627	14C1 628	14C4 629	14C4 630
14CA 631	14D6 632	14D9 633	14DA 634	14DA 636
14E8 637	1503 638	150D 639	1513 640	1519 641
151F 642	1525 643	152D 644	153B 645	154A 647
1552 649	155E 650	1566 651	156D 652	1573 653
1584 654	158D 655	1590 656	15A1 657	15A4 658
15AD 659	15CA 660	15CD 661	15D4 662	15D4 663
15D7 664	15DA 665	15E5 666	15F0 667	1605 668
1622 669	1623 670	1623 672	1626 673	1629 674
162F 675	163E 676	1641 677	1644 678	164A 679
1659 680	165C 681	166A 682	1678 683	1686 684
1690 685	1693 686	1696 687	169C 688	16AA 689
16B8 690	16CE 691	16D8 692	16DB 693	16DC 694
16DC 695	16EB 696	16EE 697	16F1 698	16F2 699
16F6 701	16FC 702	1702 703	170B 704	170C 705
1714 706	1729 707	172A 708	172D 709	1730 710
1731 715	1842 721	1842 723	1850 724	1858 725
185D 726	186E 727	187B 728	1731 729	173B 730
1744 732	1749 733	175B 734	1761 735	1764 736
1767 737	176D 738	1773 739	177D 740	1786 741
178C 742	1792 743	1798 744	179B 745	179E 746
17B4 747	17BD 748	17C3 749	17CD 750	17D6 751
17E0 752	17E9 753	17EF 754	17F9 755	1802 756

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 679
 PACIFIC GROVE, CA 93950
 SER. # _____

1808 757	180E 758	181A 759	1820 760	182C 761
182F 763	1835 764	183E 765	183E 766	1841 767
1879 768	187D 770	1882 771	1892 772	1895 773
18A5 774	18A9 775	18B0 776	18B3 777	18B4 778
19E4 781	19E4 783	19EA 784	19F8 785	1A09 786
1A10 787	1A16 788	1A27 789	1A28 790	1A36 791
1A39 792	1A3E 793	1A3F 794	1A3F 795	1A3F 796
1A46 797	1A49 798	1A4F 799	1A54 800	1A5C 801
1A5D 802	1A5D 803	1A64 804	1A75 805	1A7D 806
1A8E 807	1A93 808	1A9A 809	1AA8 810	1AA9 811
1AA9 812	1AB7 813	1AC1 814	1AF1 815	1AF8 816
1B06 817	1B10 818	1B13 819	1B19 820	1B28 821
1B29 822	1B29 823	1B29 825	1B3A 826	1B41 827
1B44 828	1B8E 829	18C1 830	18C6 831	18C9 832
18D1 833	18D7 834	18E1 835	18E6 836	18EE 837
18E1 838	18F9 839	18FC 840	1906 841	1909 842
190C 843	190F 844	1912 845	1918 846	191B 847
191F 848	1925 849	192B 850	1931 851	1936 852
193C 853	193F 854	1947 855	194A 856	196A 857
1974 859	1979 860	1988 861	198B 862	1993 863
1996 864	19A0 865	19A3 866	19A6 867	19AB 868
19A7 869	19B1 870	19B7 871	19C5 872	19C8 873
19D0 874	19DA 875	19E0 876	19E4 877	1B45 878
1B45 880	1B52 881	1B61 882	1B68 883	1B6F 884
1B79 885	1B7A 886	1B7E 888	1B9B 889	1B9E 890
1BB8 891	1BBE 892	1BDB 893	1BDE 894	1BFB 895
1BFE 896	1C1B 897	1C1E 898	1C1F 900	1C1F 901
1C27 903	1C2D 904	1C3A 905	1C3A 906	1C3B 907
1C3B 908	1C3B 909	1C42 910	1C4C 911	1C4F 912
1C57 913	1C5A 914	1C64 915	1C6E 916	1C71 917
1C8A 918	1C91 919	1C94 920	1CB1 921	1CCA 922
1CDB 923	1CDE 924	1CE8 925	1CF0 926	1CFA 927
1D02 928	1D1B 929	1D1E 930	1D24 931	1D2A 932
1D2D 933	1D37 934	1D3A 935	1D44 936	1D47 937
1D4A 938	1D4B 939	1D4B 941	1D5C 942	1D5F 943
1D66 944	1D73 945	1D8C 946	1D96 947	1D9D 948
1D9E 949	1DA4 950	1DAA 951	1DAD 952	1DB0 953
1DB1 954	1DB1 955	1DB6 956	1DC7 958	1DCA 959
1DE3 960	1DE8 961	1DE9 962	1DE9 963	1DF3 964
1DFC 965	1E04 966	1E0A 968	1E14 969	1E1E 970
1E21 971	1E26 972	1E2D 973	1E2D 974	1E37 975
1E48 976	1E4B 977	1E4C 978	1E52 979	1E58 980
1E5B 981	1E5E 982	05E2 984	05F1 985	05FC 986
0605 988	0614 989	061B 990	061E 991	062A 992
0630 993	0636 994	0644 995	064F 996	0659 997
0661 998	0666 999	066C 1000	0671 1001	0679 1002
067E 1003	0681 1004	068F 1005	06A1 1007	06AE 1008
06B3 1009	06BA 1010	06BA 1011	06C4 1012	06C4 1013
06C7 1014				

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____